中華民國專利公報【19】【12】

【11】公告編號：571186
【44】中華民國　93 (2004)　年　01 月 11 日
【51】Int. Cl.^{7}：　G06F1/20

發明　　全 5 頁

【54】名　稱：作業系統協調式熱量管理技術

【21】申請案號：091124742　【22】申請日期：中華民國 91 (2002) 年 10 月 24 日
【11】公開編號：　【43】公開日期：中華民國　年　月　日
【30】優先權：　2001/12/12　美國　10/020,848
【72】發 明 人：
巴樂斯　庫柏　BARNES COOPER
【71】申請人：
英特爾公司　INTEL CORPORATION
美國
【74】代理人：惲軼群 先生
陳文郎 先生

1　　2

[57]申請專利範圍：

1.一種用於熱量管理之方法，其特徵為包括以下步驟：
偵測出此處理器之頻率已改變以響應處理器之冷却；以及
產生中斷以響應所偵測到頻率之改變。

2.如申請專利範圍第1項之方法，其中包括提供中斷給作業系統。

3.如申請專利範圍第1項之方法，其中包括讀取處理器之表現狀態以響應此中斷。

4.如申請專利範圍第3項之方法，其中包括決定新的表現狀態。

5.如申請專利範圍第4項之方法，其中包括安排頻帶寬度分配。

6.如申請專利範圍第2項之方法，其中包括設置週期計時器。

7.如申請專利範圍第6項之方法，其中包括以週期性間隔監視處理器的溫度。

8.如申請專利範圍第1項之方法，其中包括偵測高溫或低溫中斷並讀取處理器表現狀態，以響應此高溫或低溫中斷之偵測。
9.如申請專利範圍第1項之方法，其中偵測頻率改變包括偵測處理器相位閂鎖迴路事件。
10.如申請專利範圍第 1 項之方法，其中包括使用硬體控制之調整減速。
11.一種用於熱量管理之物件，包括儲存指令之媒體而使此以處理器為主之系統能夠執行以下步驟：
偵測出處理器之頻率已經改變以響應此處理器之冷却；以及
產生中斷以響應偵測到此頻率改變。
12.如申請專利範圍第11項之物件，更包括儲存指令使得此以處理器為主的系統能夠提供中斷給作業系統。
13.如申請專利範圍第11項之物件，更包括儲存指令使得此以處理器為主的系統能夠讀取處理器之實施狀態以響應此中斷。
14.如申請專利範圍第13項之物件，更包括儲存指令使得此以處理器為主的系統能夠決定新的表現狀態。
15.如申請專利範圍第14項之物件，更包括儲存指令使得此以處理器為主之系統能夠安排頻帶寬度分配。
16.如申請專利範圍第12項之物件，更包括儲存指令使得此以處理器為主之系統能夠設立週期性計時器。
17.如申請專利範圍第16項之物件，更包括儲存指令使得此以處理器為主之系統能夠以週期性的時間間隔監視處理器之溫度。
18.如申請專利範圍第11項之物件，更包括儲存指令使得此以處理器為主之系統能夠偵測高溫與低溫中斷，並讀取處理器表現狀態以響應所偵

測到之高溫或低溫中斷。
19.如申請專利範圍第11項之物件，更包括儲存指令使得此以處理器為主之系統偵測到處理器相位閂鎖迴路事件。
20.如申請專利範圍第11項之物件，更包括儲存指令使得此以處理器為主之系統使用經硬體控制整理減速。
21.一種用於熱量管理之系統，其特徵為包括：
處理器；
溫度感測器耦合連接至該處理器；以及
儲存器用以儲存指令其使得此處理器能夠偵測出此處理器之頻率已經改變以響應處理器之冷却，並產生中斷以響應頻率改變之偵測。
22.如申請專利範圍第21項之系統，包括儲存體其儲存作業系統，該中斷是提供給作業系統。
23.如申請專利範圍第21項之系統，其中該儲存體儲存指令，其使得此處理器能夠讀取此處理器之表現狀態以回應此中斷。
24.如申請專利範圍第21項之系統，其中該處理器決定新的表現狀態。
25.如申請專利範圍第24項之系統，其中該儲存體儲存指令，其使得此處理器能夠安排頻帶寬度分配。
26.如申請專利範圍第22項之系統，其中該儲存體儲存指令，其使得此處理器能夠設立週期性計時器。
27.如申請專利範圍第26項之系統，其中該儲存體儲存指令，其使得此處理器能夠以週期性時間間隔監視處理器溫度。
28.如申請專利範圍第21項之系統，其中該儲存體儲存指令，其使得此處理器能夠偵測高溫與低溫中斷，並讀取處理器之表現狀態以響應所偵

測到高溫或低溫中斷。

29.如申請專利範圍第21項之系統，其中該儲存體儲存指令，其使得此處理器能夠偵測到處理器相位閂鎖迴路事件。

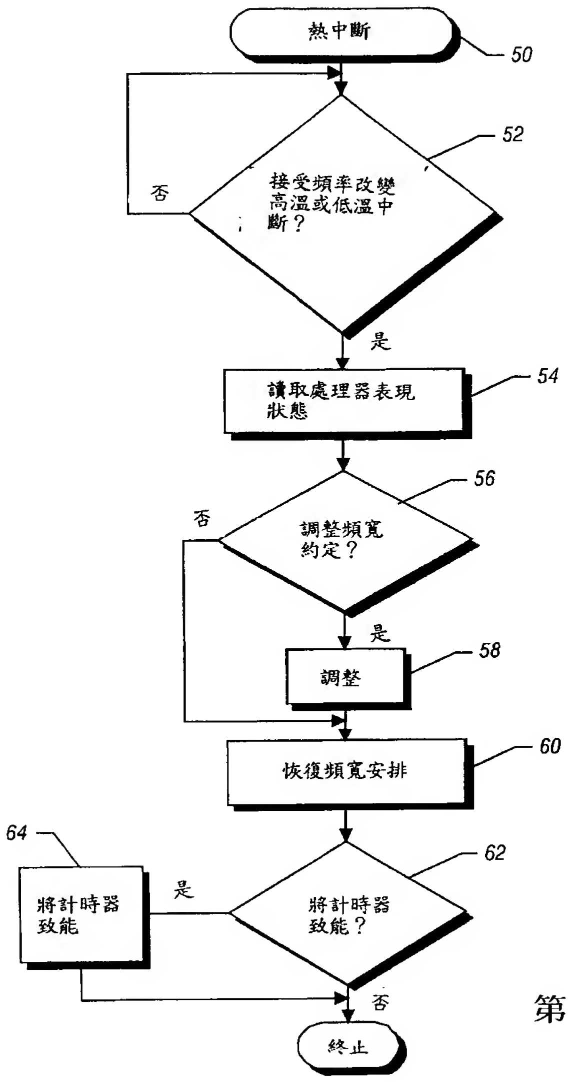
30.如申請專利範圍第21項之系統，其中包括硬體控制之調整減速。

圖式簡單說明：

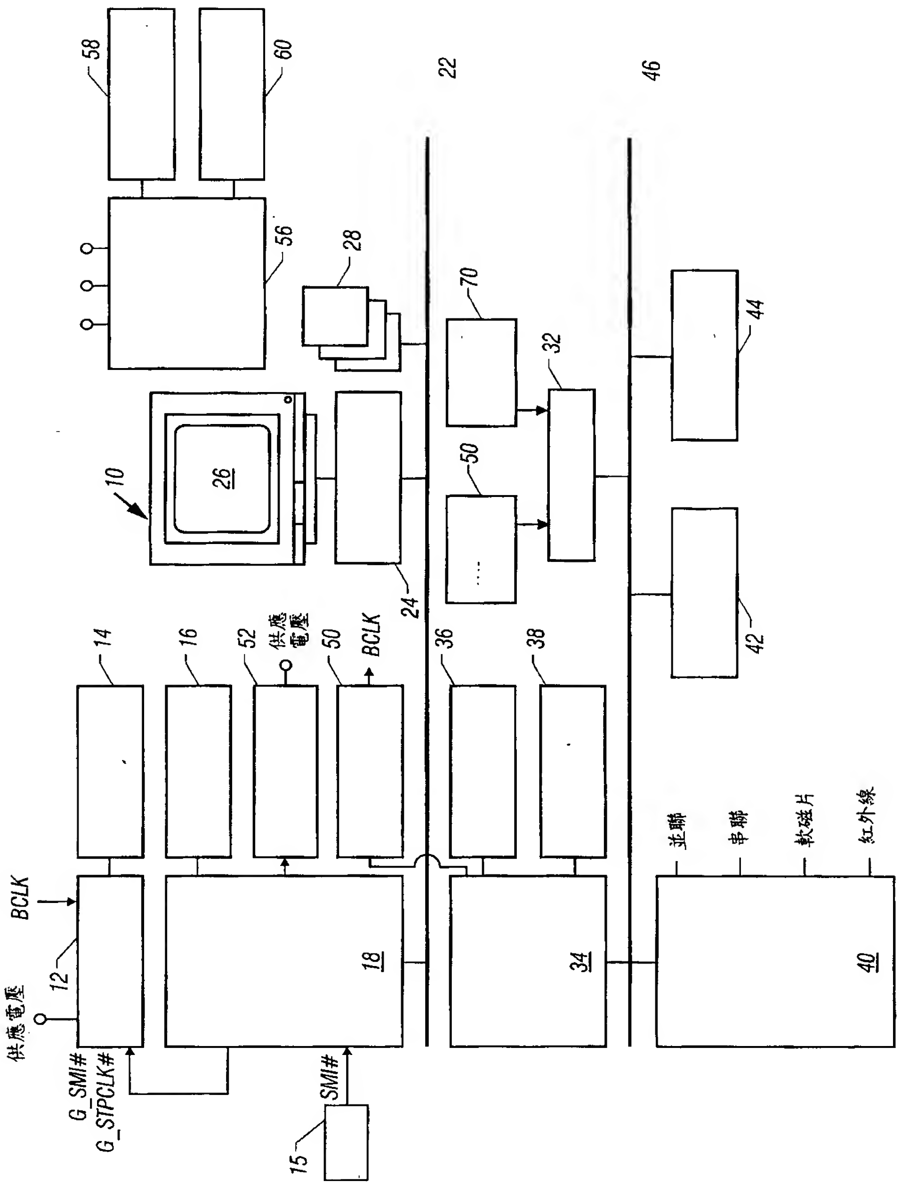
第1圖為根據本發明實施例之系統之方塊圖。

第2圖為第1圖之系統中電力管理模組之流程圖；以及

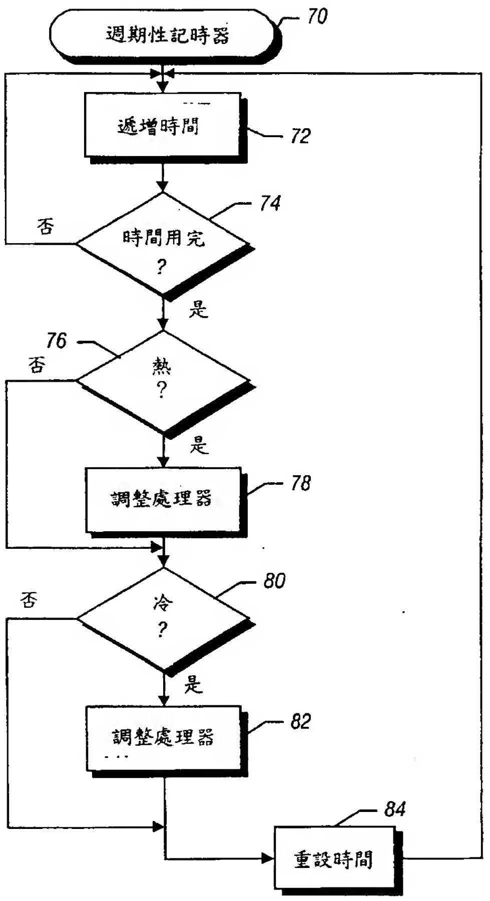
第3圖為用於第1圖系統中另外模組之流程圖。



第2圖


供應電壓
G_SMI#
G_STPCLK#
BCLK
12
15
SMI#
18
14
16
52
供應電壓
50
BCLK
36
38
34
40
並聯
串聯
軟磁片
紅外線
10
26
24
28
56
58
60
22
46
70
50
32
42
44

第 1 圖


週期性記時器
70
遞增時間
72
74
時間用完
?
否
是
76
熱
?
否
是
調整處理器
78
80
冷
?
否
是
調整處理器
82
84
重設時間

第 3 圖

P1174782—

Operating System Coordinated Thermal Management

Background

The invention relates to thermal management of processor-based systems.

Both hardware and software-controlled techniques exist for power and thermal management of processor-based systems. Software-based solutions are primarily utilized in connection with mobile platforms.

The software-controlled techniques involve an interrupt generated when a processor temperature setting is exceeded. The processor may be throttled after detecting an over temperature condition by polling processor temperature. Generally, the software-controlled solutions have a slower response time than the hardware-controlled solutions. In addition, there tends to be overshoot and undershoot problems with software-controlled solutions. The sensors utilized in software-controlled solutions are relatively slow and inaccurate. The on-die sensor (which is normally a diode) is not located on the hottest part of the processor die.

The hardware-controlled solution, used in systems other than mobile systems, involves a processor that automatically engages processor throttling, reducing the effective clock rate when a temperature condition is exceeded and disabling throttling when the processor is sufficiently cool. The hardware-controlled solution is based on an on-die binary sensor that indicates whether the processor is either hot or not hot. An interrupt capability may be available but is generally not utilized by the operating system due to the infrequency of throttling in desktop systems which are the primary applications for hardware-controlled solutions. As a result, operating systems may be unaware of hardware-controlled throttling.

The software-controlled solution is based on the premise that the platform exposes a variety of trip points to the operating system. A trip point is a temperature for a particular thermal region when some action should be taken. As the temperature goes above or below any trip point, the platform is responsible for notifying the operating system of this event and the operating system then takes an appropriate action.

When a temperature crosses a passive trip point, the operating system is responsible for implementing an algorithm to reduce the processor's temperature. It may do so by generating a periodic event at a variable frequency. The operating system then monitors the current temperature as well as the last temperature and applies an algorithm to make performance changes in order to keep the processor at the target temperature.

While current versions of hardware-controlled throttling reduce the frequency of the processor by rapidly stopping and starting the processor, future versions of hardware-controlled throttling may reduce the performance state of the processor by reducing both frequency and voltage. Because the hardware-controlled throttling is directly activated and has an extremely fast response time, the trip point for triggering the passive thermal management can be set near the high temperature specification of the processor (known as the junction temperature), thereby delivering high performance for most system designs.

Software-controlled throttling is exposed to the operating system, allowing the operating system to know the processor performance at all times. This becomes especially important with future operating systems that guarantee some quality of service based upon the processor performance to the executing applications. This concept is known as guaranteed bandwidth allocation and is based on the processor's current performance level.

Hardware-controlled throttling is advantageous in that it delivers the best possible performance in any given thermal solution, has extremely fast response time and does not throttle prematurely. A disadvantage to hardware-controlled throttling is that the operating system is completely unaware that the processor performance has been altered. Because of this, it may be expected that hardware-controlled throttling may cause issues with future operating systems that implement a guaranteed bandwidth scheduling.

Thus, there is a need for thermal management solutions that achieve advantages of both hardware and software-controlled techniques.

Brief Description of the Drawings

Figure 1 is a block diagram of a system according to an embodiment of the invention;

Figure 2 is a flow diagram of a power management module in the system of Figure 1; and

Figure 3 is a flow diagram for another module in the system of Figure 1.

Detailed Description

Referring to Figure 1, a processor-based system 10 according to an embodiment of the invention includes one or more processors 12. The system 10 may include a general- or special-purpose computer, a microprocessor- or microcontroller-based system, a hand-held computing device, a set-top box, an appliance, a game system, or any controller-based device in which the controller may be programmable.

One or more temperature sensor units 15 monitor system temperature in one or more corresponding thermal zones, each capable of issuing an interrupt, e.g., a system management interrupt (SMI), a system controller interrupt (SCI), or some other notification when a sensed temperature rises above a preset target temperature T_t or falls below the target temperature T_t.

In one embodiment, when the monitored temperature is above T_t, a thermal engage SMI is generated. On the other hand, when the monitored temperature is below T_t, a thermal disengage SMI is generated. While the monitored temperature remains above or below T_t, the thermal engage or disengage SMI may be generated at periodic intervals to allow software or firmware to manage the performance level of the processor.

In alternative embodiments, other components (e.g., bridge controller chips, peripheral controllers) in the system may be transitioned between or among the different performance states as well as throttled for system thermal management. In addition, thermal management in the system 10 may be performed independently for multiple thermal zones.

In Figure 1, the interrupt event generated by the temperature sensor unit 15 may be routed directly to the processor 12 or to a host bridge 18 coupled between the processor 12 and a system bus 22, which may in one embodiment be a Peripheral Component Interconnect (PCI) bus, as defined in the PCI Local Bus Specification, Production Version, Revision 2.1, published on June 1, 1995. Alternatively, the interrupt event may be stored as a memory or I/O-mapped register bit that is polled by a software or firmware module.

To perform throttling, a clock control input (such as the stop clock input illustrated as G_STPCLK# in Figure 1 to an 80x86 or Pentium® family processor from Intel Corporation) is activated and deactivated according to a preset duty cycle. The signal G_STPCLK# is generated by thermal management control logic and routed to the STPCLK# input pin of processors made by Intel for example. The STPCLK# internally gates clocks to the core of these processors. Activation of the clock control input (by driving G_STPCLK# low, for example) causes the processor 12 to enter a significantly reduced power mode in which an internal clock of the processor is stopped and most functions are disabled. Throttling is thus accomplished by activating the clock control input a certain percentage of the time to disable processor activity while allowing processor activity the rest of the time.

Other components of the system 10 include a clock generator 50 that generates a host clock BCLK to the processor 12 and a voltage regulator 52 that regulates the supply voltage of the processor 12. In one embodiment, the clock generator 50, processor 12, and voltage regulator 52 are controllable to transition the system 10 between or among different performance states.

A cache memory 14 is coupled to the processor 14 and system memory 16 is controlled by a memory controller in the host bridge 18. The system bus 22 may be coupled to other components, including a video controller 24 coupled to a display 26 and peripheral devices coupled through slots 28. A secondary or expansion bus 46 is coupled by a system bridge 34 to the system bus 22. The system bridge 34 may include interface circuits to different ports, including a universal serial bus (USB) port 36 (as described in the Universal Serial Bus Specification, Revision 1.0, published in January 1996) and mass storage ports 38 that may be coupled to mass storage devices such as a hard disk drive, compact disc (CD) or digital video disc (DVD) drives, and the like.

Other components coupled to the secondary bus 46 may include an input/output (I/O) circuit 40 connectable to a parallel port, serial port, floppy drive, and infrared port. A non-volatile memory 32 for storing Basic Input/Output System (BIOS) routines may be located on the bus 46, as may a keyboard device 42 and an audio control device 44. The main power supply voltages in the system 10 are provided by a power supply circuit 56 that is coupled to a battery 60 and an external power source outlet 58. References to specific components in the system 10 are for illustrative purposes--it is to be understood that other embodiments of the system 10 are possible.

Various software or firmware layers (formed of modules or routines, for example), including applications, operating system modules, device drivers, BIOS modules, and interrupt handlers, may be stored in one or more storage media in the system. The storage media includes the hard disk drive, CD or DVD drive, floppy drive, non-volatile memory, and system memory. The modules, routines, or other layers stored in the storage media contain instructions that when executed causes the system 10 to perform programmed acts.

The software or firmware layers, such as the thermal interrupt software 50 and the periodic timer software 70, can be loaded into the system 10 in one of many different ways. For example, code segments stored on floppy disks, CD or DVD media, the hard disk, or transported through a network interface card, modem, or other interface mechanism may be loaded into the system 10 and executed as corresponding software or firmware layers. In the loading or transport process, data signals that are embodied as carrier waves (transmitted over telephone lines, network lines, wireless links, cables, and the like) may communicate the code segments to the system 10.

Thermal interrupt software 50 initially determines whether a frequency change, high temperature or a low temperature interrupt has been received as indicated in diamond 52. High temperature and low temperature interrupts are conventional software-controlled interrupts. The frequency change interrupt is hardware-controlled but differs from conventional hardware-controlled interrupts in that the operating system is notified at an appropriate time, for example to enable guaranteed bandwidth allocation.

In some systems, rather than simply throttle the processor, the performance state of the processor 12 may be directly controlled. The performance state involves both the frequency and the voltage. In such case, throttling may directly reduce or increase the performance state as the processor 12 goes above or below the on-die sensor 15 trip point.

On transitions to a lower performance state (due to the processor getting hotter), the processor's frequency is reduced prior to reducing the processor voltage. The processor's performance, as seen by the operating system, will be reduced immediately. That is, the performance reduces as soon as the frequency is reduced.

On transitions to a higher performance state (due to the processor cooling down), the processor's frequency is not increased until the voltage is changed to a higher level. This voltage change is dependent on many factors. In general, it takes some amount of time to create the voltage change. As a result, the performance change would lag the interrupt event if the interrupt event were generated upon the voltage change.

Instead, the interrupt event may be generated any time the processor's phase locked loop (PLL) relocks at a new frequency level. Thus, when the interrupt fires, the operating system can read the processor's performance state, determine the new performance level of the processor, reschedule guaranteed bandwidth allocations as required and then resume normal operations.

Referring to Figure 2, if an event is detected in diamond 52, the processor performance state is read. This may be done by accessing processor registers to determine the cause of the event as well as to take further action. The amount of code is small, bounded and can be page locked in physical memory in some embodiments.

When the operating system receives any of the three sources of thermal management interrupt vectors, as determined in diamond 52, the processor can check whether the processor is hot or cold and look up the current performance state, as indicated in block 54, based upon registers defined already and take appropriate action. Typical registers may include on-die throttling control and the performance state status register.

In accordance with one embodiment of the present invention, the new interrupt may be added to existing interrupt models for hot and cold interrupt generation. The frequency change interrupt may have an enable bit to allow the operating system to enable or disable the event, but no status register may be needed in some embodiments.

Next, a check at diamond 56 determines whether the bandwidth contracts need to be adjusted in view of the current processor performance state. If so, the contracts are adjusted as indicated in block 58. Thereafter, the bandwidth scheduling may be resumed as indicated in block 60. A check at diamond 62 determines whether a periodic timer should be implemented. The operating system may enable a periodic timer event to begin monitoring the processor temperature if the interrupt is indicative of a processor thermal event, as indicated block 64.

The periodic timer software 70, shown in Figure 3, begins by incrementing the time as indicated in block 72. A check at diamond 74 determines whether a time out has occurred. If so, a check at diamond 76 determines whether the processor is still hot. If so, the operating system may decide to reduce the processor performance state and/or enable on-die throttling and/or increase the internal effective frequency of on-die throttling as indicated in block 78.

A check at diamond 80 determines whether the processor has now cooled off. If so, the operating system may decide to increase the processor performance state and/or disable on-die throttling and/or increase the internal effective frequency of on-die throttling as indicated in block 82. Thereafter, the time is reset as indicated in block 84 and the flow may recycle.

Particularly with mobile platforms, increased performance may be realized by utilizing the software and hardware-controlled solutions described above. By allowing hardware-controlled throttling to coexist with operating system dispatch algorithms, fast, efficient thermal management may be achieved in some embodiments while still enabling guaranteed bandwidth allocation schemes.

While the present invention has been described with respect to a limited number of embodiments, those skilled in the art will appreciate numerous modifications and variations therefrom. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of this present invention.

What is claimed is:

1. A method for thermal management comprising:

detecting that a processor's frequency has changed in response to processor cooling; and

generating an interrupt in response to the detection of the frequency change.

2. The method of claim 1 including providing an interrupt to an operating system.

3. The method of claim 1 including reading the performance state of the processor in response to the interrupt.

4. The method of claim 3 including determining a new performance state.

5. The method of claim 4 including scheduling a bandwidth allocation.

6. The method of claim 2 including setting up a periodic timer.

7. The method of claim 6 including monitoring the processor temperature at periodic intervals.

8. The method of claim 1 including detecting a high temperature or a low temperature interrupt and reading the processor performance state in response to the detection of a high temperature or a low temperature interrupt.

9. The method of claim 1 wherein detecting a frequency change includes detecting a processor phase locked loop event.

10. The method of claim 1 including using hardware controlled throttling.

11. An article for thermal management comprising a medium storing instructions to enable a processor-based system to:

detect that a processor's frequency has changed in response to processor cooling; and

generate an interrupt in response to the detection of the frequency change.

12. The article of claim 11 further storing instructions to enable a processor-based system to provide an interrupt to an operating system.

13. The article of claim 11 further storing instructions to enable a processor-based system to read the performance state of the processor in response to the interrupt.

14. The article of claim 13 further storing instructions to enable a processor-based system to determine a new performance state.

15. The article of claim 14 further storing instructions to enable a processor-based system to schedule a bandwidth allocation.

16. The article of claim 12 further storing instructions to enable a processor-based system to set up a periodic timer.

17. The article of claim 16 further storing instructions to enable a processor-based system to monitor the processor temperature at periodic intervals.

18. The article of claim 11 further storing instructions to enable a processor-based system to detect a high temperature or a low temperature interrupt and read the processor performance state in response to the detection of a high temperature or a low temperature interrupt.

19. The article of claim 11 further storing instructions to enable a processor-based system to detect a processor phase locked loop event.

20. The article of claim 11 further storing instructions to enable a processor-based system to use hardware controlled throttling.

21. A system for thermal management comprising:

a processor;

a temperature sensor coupled to said processor; and

a storage storing instructions that enable the processor to detect that the processor's frequency has changed in response to processor cooling and generate an interrupt in response to detection of the frequency change.

22. The system of claim 21 including a storage storing an operating system, said interrupt being provided to the operating system.

23. The system of claim 21 wherein said storage stores instructions that enable the processor to read the performance state of the processor in response to an interrupt.

24. The system of claim 21 wherein said processor determines a new performance state.

25. The system of claim 24 wherein said storage stores instructions that enable the processor to schedule a bandwidth allocation.

26. The system of claim 22 wherein said storage stores instructions that enable the processor to set up a periodic timer.

27. The system of claim 26 wherein said storage stores instructions that enable the processor to monitor the processor temperature at periodic intervals.

28. The system of claim 21 wherein said storage stores instructions that enable the processor to detect a high temperature or a low temperature interrupt and read the processor performance state in response to the detection of a high temperature or a low temperature interrupt.

29. The system of claim 21 wherein said storage stores instructions that enable the processor to detect a processor phase locked loop event.

30. The system of claim 21 including hardware controlled throttling.

Operating System Coordinated Thermal Management

Abstract of the Disclosure

A processor's (12) performance state may be adjusted based on processor temperature. On transitions to a lower performance state due to the processor (12) getting hotter, the processor's frequency is reduced prior to reducing the processor voltage. Thus, the processor's performance, as seen by the operating system, is reduced immediately. Conversely, on transitions to a higher performance state, due to the processor cooling down, the processor's frequency is not increased until the voltage is changed to a higher level. An interrupt event may be generated anytime the processor's phase locked loop relocks at a new frequency level. Thus, when the interrupt fires, the operating system can read the processor's performance state. As a result, interrupts are not generated that would cause processor performance to lag the interrupt event.

P11247F-2

1/3

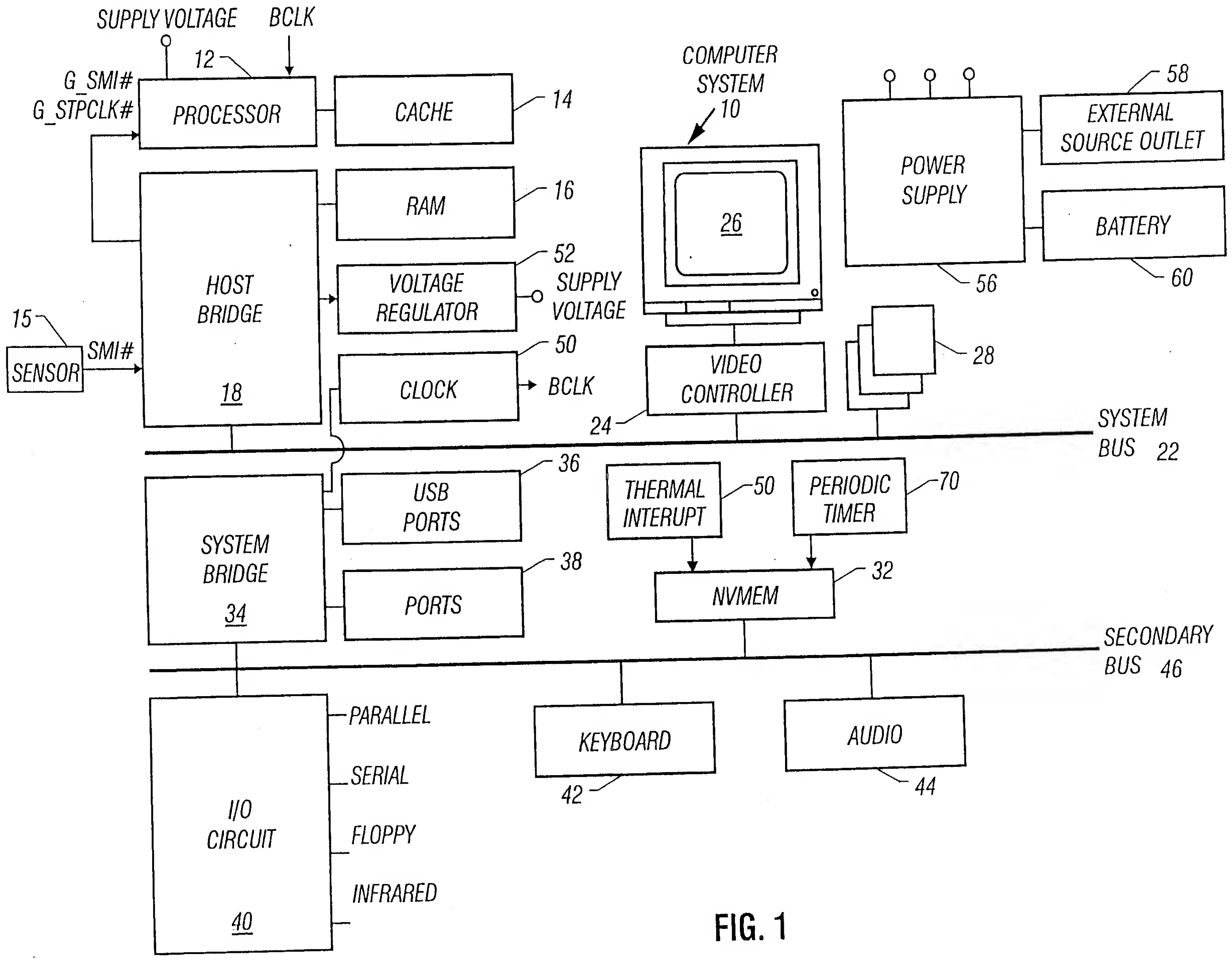


FIG. 1

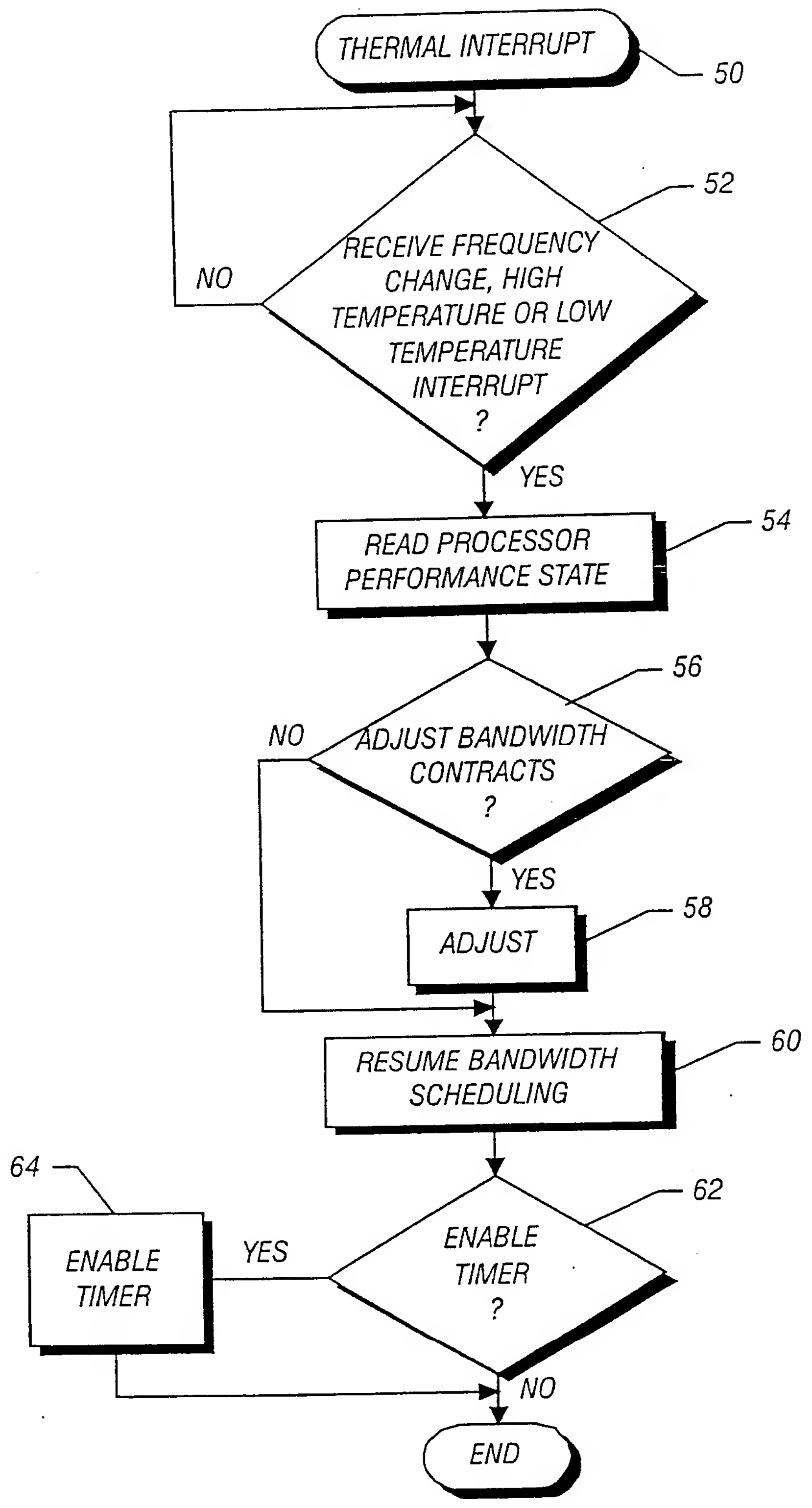

THERMAL INTERRUPT
50
RECEIVE FREQUENCY CHANGE, HIGH TEMPERATURE OR LOW TEMPERATURE INTERRUPT ?
52
NO
YES
READ PROCESSOR PERFORMANCE STATE
54
ADJUST BANDWIDTH CONTRACTS ?
56
NO
YES
ADJUST
58
RESUME BANDWIDTH SCHEDULING
60
ENABLE TIMER ?
62
YES
NO
ENABLE TIMER
64
END

FIG. 2

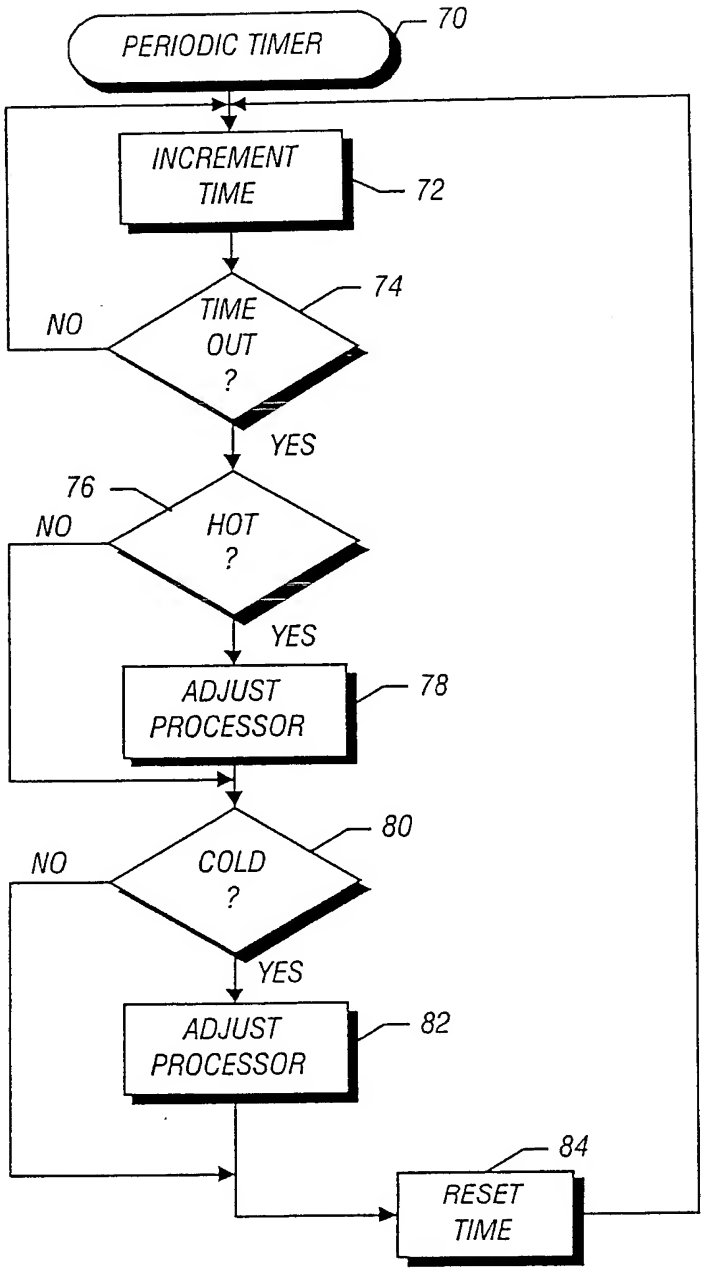

PERIODIC TIMER
70
INCREMENT TIME
72
TIME OUT ?
74
NO
YES
76
HOT ?
NO
YES
ADJUST PROCESSOR
78
COLD ?
80
NO
YES
ADJUST PROCESSOR
82
RESET TIME
84

FIG. 3